Dynamic file allocation on a non-shared disk cluster

Arie Shoshani and Doug Olson

January, 2001

The problem

In data intensive applications, the bottleneck is in getting the data to the processing system. Typically, the amount of processing per unit of data is small. In HENP applications, the analysis of the data can be trivially processed in parallel, since each event can be processed independently of other events. If all the files needed for processing reside on a single disk file system that is shared by all the processors, then the file system becomes the bottleneck.

Therefore a different architecture is now being developed, where each workstation has it own reasonably large disk system. These disk systems are not shared. To execute a parallel job, the data needs to be distributed to the various disk systems, in such a way as to maximize the parallel processing on the system.

Several assumptions are made. 1) The archived files are on tape, and managed by a mass storage system, such as HPSS. 2) Only a fraction of the files on tape get replicated on the various disks depending on the access patterns of the analysis programs. 3) Once a file is in a certain disk, all processing for events in this file occur on the processor attached to this disk. Additional requests to process a file are directed to that processor. 4) Files are not restructured or partitioned. That is, entire files are replicated. 5) Files contain multiple events. Events are distributed on files according to the order they are produced, and therefore are random relative to their properties (such as energy, number of particles produced, etc.) 6) The cost of processing the data from a file relative to moving the file to that processor's disk is very small.

The problem is how to allocate the files to the different disks.

A proposed solution: dynamic file allocation

Given the assumptions above, for a job (such as an analysis program) that requires N files, the ideal solution is to allocate the N files over M processors in an equal way. This can be achieved by simple round robin allocation. Note, that files can have varying number of events that need to be processed for a given job. However, because of assumption 5 and 6, we assume that processing of each file by a processor is roughly the same.

The file allocation can be decided ahead of time by predicting the access patterns. This requires periodic review of how well the predictions match reality, and re-allocating files accordingly. This approach has several disadvantages. 1) Accurate access pattern predictions are hard to make. Thus, the system may be unbalanced for the job mix currently being processed. 2) Event if predictions are accurate, the job mix is likely to change over time, requiring a restructuring of the allocation, which can be a major disruption of the system. 3) It is necessary to have human intervention. And an administrator managing the file allocation when the system is unbalanced.

Therefore, we propose a dynamic file allocation solution. It is based on the concept of a local Disk Resource Manager (DRM) on each processing unit, and a Parallel Job Dispatcher (PD) that controls each job. Next is a short description on what each performs.

A DRM keeps track of what is on its disk, and whether a file is currently used by one or more process. When space is needed to bring a new file into the DRM's disk, the DRM is responsible to make the decision which file(s) to remove. A file cannot be removed when it is in use, i.e. being accessed by one or more processes. The DRM manages its own disk according to a "file purging policy". A simple-minded policy can be based on removing the file that was Least-Recently-Used (LRU). More sophisticated policies can be based on history of usage. Finally, a request to bring a file into a DRM's cache can be made to that

DRM, and it will initiate the file transfer from the HPSS system, monitor its progress, and notify the requester (in this scenario, it is the PD) when the file transfer completed.

The PD works as follows. When a job is scheduled, all the files needed are known. The PD checks with a file-disk catalog which files are already in any of the disks. (We discuss later how it is maintained). For these files, it notifies the corresponding DRMs that the files are needed (i.e. a request to "pin" the file), and it assigns the processing necessary. For the remaining files it determines the location of the disk that each file should be transferred to. It then requests each DRM to get the its assigned file from HPSS. For each file that was processed, the PD sends a "release file" to the corresponding DRM.

The file-disk catalog (FDC) is updated dynamically by the PD. It marks a file that it requested to be transferred, so another job can issue a transfer request for the same file to the same DRM. It marks the file as being in disk when the DRM notifies it that the file transfer was completed. When some file is removed from a disk to make space, the DRM notifies the PD requester, and it updates the FDC.

This design is made so that the PD and the DRMs are completely independent modules. If one of the processors crashes, the system can continue to work, since only that disk is affected. If some local DRM does not respond, the PD can request a file that resided on that disk to be transferred to HPSS to another disk.

The solution described above assumes a single replica per file, i.e. the replica resides only on a single disk. A more general solution can permit multiple replicas of files that are very popular to be made. Such an algorithm will be developed in the future. Also, this architecture can permit multiple PDs to run, as long as they all access the same FDC.

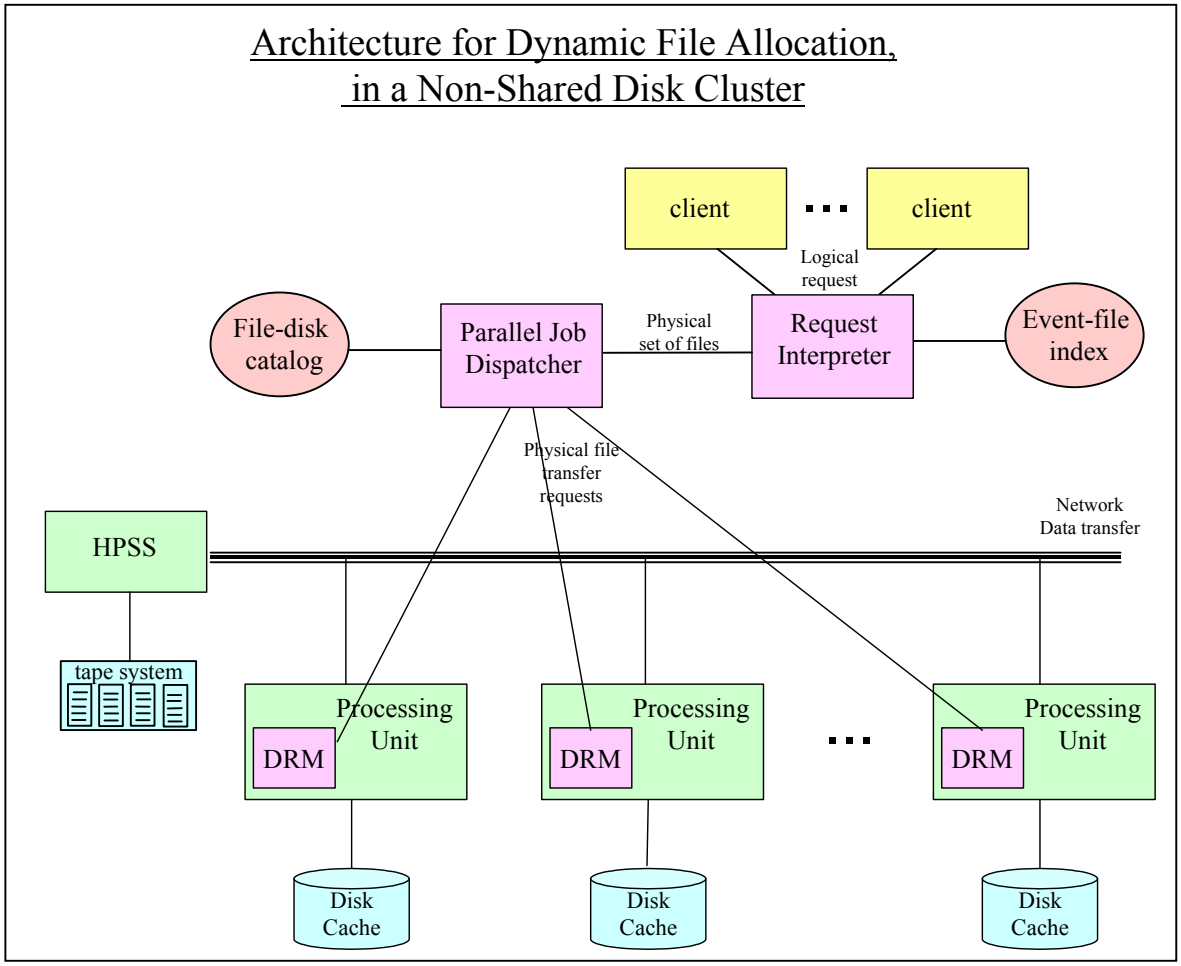
The architecture

The figure below is a schematic diagram of the components needed to support the dynamic allocation solution. The components in purple are needed. As can be seen, it includes a Request Interpreter component that accepts a "logical request" based on properties of the events (such as energy and momentum), and determines the physical files that need to be processed. For each such job, the Parallel Job Dispatcher determines if some of the files are in some disk. If they are not, it communicates with the DRMs where it wishes the missing files to reside. No communication with HPSS is necessary since the DRMs manage the file access from HPSS directly.

Coordination with other sciDAC projects

This proposal build on technology that was developed by previous projects and will be supported as products of other proposed sciDAC projects. Specifically, the Request Interpreter component was initially developed under the HENP-GC project, and is being proposed as one of the components of the Scientific Data Management Enabling Technology Center (SDM-ETC). Under SDM-ETC this components will be enhanced to be available as a stand-alone component with various enhancements. In addition, the initial versions of the DRM component are being developed under the PPDG project, and will be further developed under the Storage Resource Management (SRM) proposal for Collaboratories Middleware Technology.

The main components to be developed under this proposal are the Parallel Job Dispatcher (PD) and the File-Disk Catalog (FDC).


Architecture for Dynamic File Allocation,
in a Non-Shared Disk Cluster
client
client
Logical
request
File-disk
catalog
Parallel Job
Dispatcher
Physical
set of files
Request
Interpreter
Event-file
index
Physical file
transfer
requests
Network
Data transfer
HPSS
tape system
Processing
Unit
DRM
Processing
Unit
DRM
Processing
Unit
DRM
Disk
Cache
Disk
Cache
Disk
Cache